Learning – process by which system improves performance from experience.

Learning – study of algorithms that improve their performance with experience.

Supervised learning   
setup - A model is trained using labeled data, where the input is paired with the correct output labels, with the objective of learning to predict the output label, given an input data example.

Learning – By analyzing a training set of labelled examples and adjusting model parameters in order to minimize the error between model’s predictions and correct output labels.

Drawbacks

Labelled data required

Susceptible to bias – if training data biased ,model will inherit these biases. Ex- if a dataset for facial recognition is predominantly composed of faces from certain ethnic groups, the model might perform poorly on faces from underrepresented groups.

Generalization issues - Models might struggle to generalize to new, unseen data. This is especially true if the training data is not representative of the real-world scenarios in which the model will be used. For example, a model trained on images from one geographic location might not perform well on images from a different location with distinct environmental characteristics.

May require feature engineering - This is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data. Feature engineering is often a crucial step because the right set of features can significantly improve model performance, while the wrong set can hinder it.

Unsupervised learning

Setup - Model trained using unlabelled data with objective of finding underlying hidden structures and patterns in the data.

Learning – By analyzing training set of unlabelled examples and adjusting model parameters in order to maximize learning objective.

* **Lack of Ground Truth:** Without labeled data, it's difficult to know if the model's findings are meaningful or accurate. There's no straightforward way to validate the model's output, as there's no "correct answer" to compare it with.
* **Difficult to Evaluate:** Evaluating the performance of an unsupervised model is challenging. Metrics that are used in supervised learning, like accuracy or precision, are not applicable. Instead, alternative methods, such as silhouette scores for clustering or the elbow method, are used, but these might not always provide a clear indication of model performance.
* **Curse of Dimensionality:** Unsupervised learning can suffer from the curse of dimensionality, especially in clustering tasks. As the number of dimensions (features) increases, the data becomes sparser, making it difficult to find meaningful clusters or patterns.
* **Computationally Intensive:** Many unsupervised learning algorithms require significant computational resources, especially for large datasets with many features. This computational demand can be a limiting factor in terms of scalability and practicality.

Unsupervised learning is valuable for exploratory data analysis, discovering hidden patterns, or data preprocessing, but these drawbacks highlight the importance of careful method selection, algorithm tuning, and validation techniques to achieve meaningful results.

Semi-supervised Learning

Setup: a model is trained using both labelled and unlabeled data, with the objective of learning to predict the output label given an input data example.

Learning: by iteratively training the model using the most confident predictions, and augmenting the training dataset.

Semi-supervised learning is a hybrid approach that combines elements of both supervised and unsupervised learning.

It's particularly useful in scenarios where obtaining labeled data is expensive or labor-intensive, but there's an abundance of unlabeled data. Here's an overview of its setup, learning process, and inherent drawbacks:

1. **Setup in Semi-Supervised Learning:**
   * **Combining Labeled and Unlabeled Data:** A model is trained using a mixture of labeled and unlabeled data. The labeled data provides a basis for learning the desired output, while the unlabeled data helps in understanding the broader structure of the input space.
   * **Objective:** The primary goal remains similar to that in supervised learning: to predict the output label for a given input. However, the inclusion of unlabeled data aims to enhance the learning process, making the model more robust and possibly more generalizable.
2. **Learning Process:**
   * **Iterative Training:** The model initially trains on the labeled data. It then makes predictions on the unlabeled data. The most confident of these predictions are often treated as 'pseudo-labels' and used to augment the training dataset.
   * **Self-training and Refinement:** This process is typically iterative. The model refines its understanding and predictions in each iteration, progressively incorporating more of the unlabeled data into the training process.

Despite its advantages, semi-supervised learning faces several challenges:

* **Sensitive to Data Quality:** The quality of both labeled and unlabeled data is crucial. Poor-quality data, especially in the labeled subset, can significantly impair the model's learning.
* **Difficult to Evaluate:** Evaluating semi-supervised models can be challenging due to the use of unlabeled data. It's harder to ascertain the model's accuracy and generalization capability without a comprehensive set of labeled test data.
* **Difficult to Interpret:** The iterative process of augmenting training data with pseudo-labels can make these models more complex and less interpretable. Understanding why a model makes certain predictions can be challenging, especially when those predictions are based partly on data that the model itself has labeled.
* **Computationally Intensive:** The iterative nature of semi-supervised learning, along with the need to handle larger datasets (including both labeled and unlabeled data), can make these models computationally demanding.

In summary, semi-supervised learning offers a valuable middle ground when both labeled and unlabeled data are available. However, it requires careful handling of data quality, model evaluation, and computational resources to be effective.

Top of Form

Semi-supervised learning

Self-learning

Pick a small amount of labeled data and use this dataset to train a base model.

Then apply the process known as pseudo-labeling by taking the partially trained model and use it to make predictions for the unlabeled data.

Take the most confident predictions made by the model and add them into the labeled dataset; hence creating a new, combined input to train an improved model.

Repeat the process for several iterations.

It’s important to monitor the model’s performance throughout this process. This can be done using a validation set, or through other performance metrics.

The process is typically stopped once the model’s performance plateaus or begins to decrease, indicating that adding more pseudo-labeled data is no longer beneficial.

Co-learning

• Train two individual classifiers based on two views of data.

• Views: different sets of features but same instances.

• The bigger pool of unlabeled data receives pseudo-labels from both classifiers.

• Classifiers co-train one another using the pseudo-labels with the highest confidence level.

• The predictions of the two classifiers are combined.

Co-learning, also known as co-training, is a semi-supervised machine learning technique that leverages two different views of the same data. This method is particularly effective when each view is sufficient to train a good classifier but may be complementary when combined. Here's an overview of the co-learning process:

1. **Training Individual Classifiers:**
   * Initially, two separate classifiers are trained. Each classifier is based on a different "view" of the data.
   * A "view" refers to a different set of features. For example, in a text classification task, one view could be based on the text's syntactic features and the other on semantic features.
2. **Views on the Same Instances:**
   * Despite having different sets of features, both views pertain to the same set of instances. This means that each part of the data is described in two different ways.
3. **Utilizing a Larger Pool of Unlabeled Data:**
   * There is typically a larger pool of unlabeled data available. Both classifiers are used to assign pseudo-labels to this data.
   * Each classifier makes predictions independently based on its view of the data.
4. **Co-Training Using Pseudo-Labels:**
   * The classifiers co-train one another using the pseudo-labels they generate. The key here is to use the pseudo-labels with the highest confidence level.
   * This process involves each classifier selecting the most confident predictions it has made and sharing these pseudo-labels with the other classifier. This allows each classifier to learn from the other's strengths.
5. **Combining Predictions:**
   * The predictions of the two classifiers can be combined to make a final decision. This could be done through various methods such as voting, averaging the probabilities, or more complex decision-making strategies that consider the confidence levels of each classifier.
6. **Iterative Improvement:**
   * The process is iterative. With each cycle, the classifiers retrain on their original labeled data augmented by the new pseudo-labeled data from the other classifier.
   * Over iterations, this can lead to improved performance as each classifier benefits from the additional, albeit pseudo-labeled, training data.
7. **Monitoring and Validation:**
   * Throughout the co-learning process, it's important to monitor the performance of each classifier and the combined system.
   * Periodic validation using a labeled dataset is crucial to ensure that the addition of pseudo-labels is improving rather than degrading the performance.

Co-learning leverages the strengths of multiple views of data and the concept of semi-supervised learning to improve model performance, especially in scenarios where labeled data is scarce but unlabeled data is abundant. The success of co-learning depends on the complementarity of the views and the careful management of the pseudo-labeling process.

The statement "The bigger pool of unlabeled data receives pseudo-labels from both classifiers" in the context of co-learning refers to a key step in the semi-supervised learning process where two classifiers, each trained on a different view (set of features) of the same instances, are used to generate labels for a large set of data that initially has no labels. Let's break down this process:

1. **Two Classifiers, Two Views:**
   * In co-learning, you have two classifiers. Each classifier is trained on a distinct "view" of the data. For example, in a text analysis task, one classifier might be trained on word frequency data (View 1) and another on the syntactic structure of the sentences (View 2). Though the features (views) are different, both classifiers are trained on the same set of instances (e.g., the same set of documents).
2. **Large Pool of Unlabeled Data:**
   * Apart from the initially labeled data used for training, there is a larger pool of data that doesn't have any labels. This is the data that the model needs to learn from, despite the absence of explicit guidance in the form of labels.
3. **Generating Pseudo-Labels:**
   * Each classifier, after being trained on its respective view of the labeled data, is then used to make predictions on this larger, unlabeled dataset.
   * These predictions are known as "pseudo-labels" because they are not true labels provided by human annotators but are inferred by the classifiers.
4. **Pseudo-Labels from Both Classifiers:**
   * Both classifiers independently assign labels to the unlabeled data based on their learning and perspective. Since each classifier has been trained on different aspects of the data (different views), they might have different strengths and insights.
5. **Combining Pseudo-Labels:**
   * The next step involves combining these pseudo-labels to enrich the training process. There are various ways to do this, such as taking the label on which both classifiers agree, or using a confidence threshold to select the most reliable labels from each classifier.
6. **Iterative Process:**
   * These pseudo-labels are then used to further train each classifier, not just on its original view but potentially also incorporating insights from the other classifier's view. This iterative process continues, with each classifier contributing to and learning from the pseudo-labeled data, improving over time.

In essence, this statement highlights a collaborative effort between two different machine learning models to expand their knowledge and improve their accuracy using a large dataset that initially lacks labels. This collaboration allows each model to benefit from the other's strengths and compensates for its weaknesses, potentially leading to more robust and accurate predictions.

Top of Form

In the co-learning (or co-training) approach, the pseudo-labels generated by one classifier are added to the training data of the other classifier.

1. **Initial Training on Separate Views:**
   * Each classifier is initially trained on a different view of the data. View 1 might contain one subset of features, while View 2 contains a different subset. Despite the feature differences, both classifiers work on the same instances.
2. **Generation of Pseudo-Labels:**
   * After this initial training phase, each classifier is used to make predictions on the unlabeled data. These predictions are the pseudo-labels. They are called "pseudo" because they are not verified by a human annotator but are generated based on the model's current understanding and learning.
3. **Cross-Training with Pseudo-Labels:**
   * The pseudo-labels generated by Classifier A (trained on View 1) are then added to the training set of Classifier B (trained on View 2), and vice versa.
   * This means that each classifier is enriched with additional training data, which includes not just the features it was originally trained on but also the insights (in the form of pseudo-labels) gained from the other classifier's perspective.

Active learning

Setup: a model is trained using both labelled and unlabeled data, with the objective of learning to predict the output label given an input data example.

Learning: by iteratively training the model using a human expert annotator for the least confident predictions.

Drawbacks:

• Sensitive to data quality

• Dependence on human annotator

• Computationally intensive

Active learning is a special case of semi-supervised learning that focuses on efficiently training a model by involving a human annotator to label the most informative data points. This approach is particularly useful when labeled data is scarce or expensive to obtain. Here's an overview of its setup, learning process, and inherent drawbacks:

1. **Setup in Active Learning:**
   * **Combination of Labeled and Unlabeled Data:** The process starts with a model trained on a small amount of labeled data, with a larger pool of unlabeled data available.
   * **Objective:** The goal is similar to other machine learning tasks — to predict the output label for a given input. However, active learning specifically aims to achieve this with as few labeled instances as possible by intelligently selecting the most informative data points.
2. **Learning Process:**
   * **Iterative Training with Human Involvement:** The model is trained on the initial labeled data and then makes predictions on the unlabeled data.
   * **Selecting Data for Annotation:** Instead of using all the unlabeled data, active learning involves selecting specific instances where the model is least confident. These instances are believed to be the most informative if labeled.
   * **Involvement of Human Annotator:** These selected instances are then presented to a human expert who provides the labels.

* Once these instances are annotated, they are added to the training dataset.
* The training set now comprises the original labeled data plus the newly labeled instances.
  + **Model Refinement:** The model is retrained with this newly labeled data, improving its performance with each iteration. The cycle of prediction, selection, annotation, and retraining continues.

1. **Drawbacks of Active Learning:**
   * **Sensitive to Data Quality:** The effectiveness of active learning heavily depends on the quality of the data, especially the newly labeled instances. Poor quality or mislabeled data can lead to incorrect learning.
   * **Dependence on Human Annotators:** Active learning relies on human experts to label the data. This can be a bottleneck due to the time, cost, and availability of experts. Also, the process might be subject to human error or inconsistency.
   * **Computationally Intensive:** The iterative nature of active learning, combined with the need to continually retrain the model and evaluate which data points should be labeled next, can be computationally demanding. This is especially true for large datasets or complex models.

Active learning is particularly beneficial in scenarios where acquiring labeled data is expensive or time-consuming, and it helps to build effective models with a smaller amount of labeled data. However, its reliance on human annotators and computational demands are significant considerations that impact its applicability and efficiency.

Reinforcement learning: an agent learns to interpret its environment and optimize a given objective by interacting with the environment through actions that are either rewarded or penalized.

Federated learning: training data is locally stored in interconnected nodes, and it and cannot be shared. A central model is learned using the distributed local data by only sharing model parameters and not the data points themselves.

Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with its environment. This learning paradigm is distinct from supervised and unsupervised learning, focusing on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

Federated learning is a machine learning approach designed to train algorithms across decentralized devices or servers holding local data samples, without exchanging them. This method is particularly useful in scenarios where data privacy, confidentiality, and access rights are a concern.

Given data from distributed hospitals learn a central model that can propose the optimal patient treatment without sharing any data.

Classification is the task of learning a target function f that maps an input (attribute set) X to a class label Y.

Regression is the task of learning a target function f that maps an input (attribute set) X to a real value Y in a given range of real values.

Underfitting: an ML model is unable to capture the relationship between the input and output variables accurately, generating a high error rate on both training and test sets

Overfitting: an ML model is unable to generalize, hence the relationship it learns between the input and output variables in the training set does not reflect the true relationship in the test set, generating a low error rate on the training set but a high error rate on the test set.

A graph of a training error

Description automatically generated

1. **U-Shape Effect Graph:**
   * The graph shows two curves: one for training error and one for test error, as a function of model complexity (denoted by "f").
   * As model complexity increases, the training error (blue curve) tends to decrease because the model fits the training data better. However, if the model becomes too complex, it may start to fit the noise in the training data, rather than the underlying trend (overfitting).
   * The test error (red curve) typically follows a U-shaped curve. Initially, as complexity increases, the error decreases because the model is learning the underlying patterns in the data. After a certain point, as the model complexity continues to increase, the test error starts to increase due to overfitting.
2. **Bias-Variance Tradeoff:**
   * The graph illustrates the bias-variance tradeoff, a fundamental concept in machine learning. The area on the left where the test error is high corresponds to "High Bias and Low Variance." This is where the model is too simple and cannot capture the complexity of the data (underfitting).
   * The area on the right, where the test error starts increasing again, corresponds to "Low Bias and High Variance." This is where the model is too complex and is sensitive to the fluctuations in the training data (overfitting).

Simple models: make many assumptions on the underlying

data distribution (high bias), but they are not that sensitive to

the training set (low variance)

Complex models: make few (or no) assumptions on the

underlying data distribution (low bias), but they may end up

being very sensitive to the training set (high variance)

Hence, we want models with:

Low variance: they should not be sensitive to the training set, they should not

overfit

Low bias: they should not make unrealistic assumptions, they should be as

simple as possible, they should not under-learn.

Challenge: obtain models that achieve a good trade-off between

variance and bias

High variance in the context of machine learning refers to a model's tendency to learn too much from the training data, including the noise and the random fluctuations, which often results in a lack of generalizability to new, unseen data.

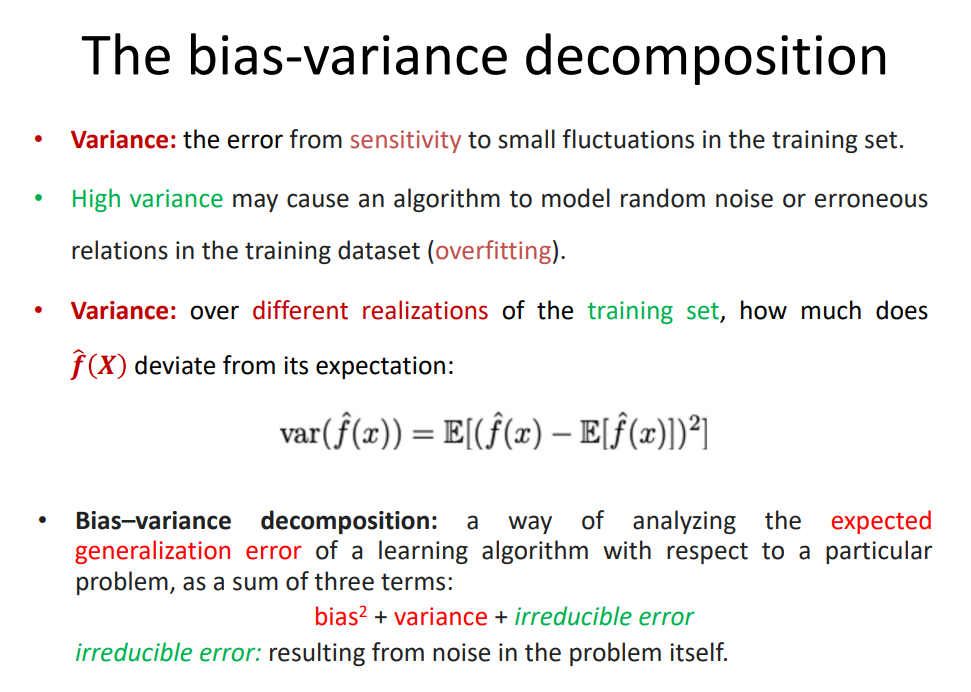
High bias – A high bias situation occurs when a model is too simple to capture the complexity of the data.

A white text with black text and red and blue text

Description automatically generated

A screenshot of a math problem

Description automatically generated



Model selection – estimating performance of different models in order to choose the best one.

Model assessment – having chosen a model ,estimating its prediction error(generalization error) on new data.

Hold out & subsampling

• Holdout (or using a validation set) – reserve 50% for training, 25% for validation, and 25% for testing

• Random subsampling – repeated holdout

• Stratified subsampling – preserve the class balance in the samples

Partition the N data samples into k disjoint subsets

train on k-1 partitions – test on the remaining one

Each subset is given a chance to be in the test set

Performance is averaged over the k iterations.

Learning curve – plot that shows changes in learning performance (y-axis) over size of training set (x-axis).

Performance improves as we move to 100 examples.

Increasing to 200 – very small benefit.

Training set =200 examples

For 5 fold CV , training set =size 160 same as performance for 200 examples

Hence no bias.

A graph of loss and loss

Description automatically generated

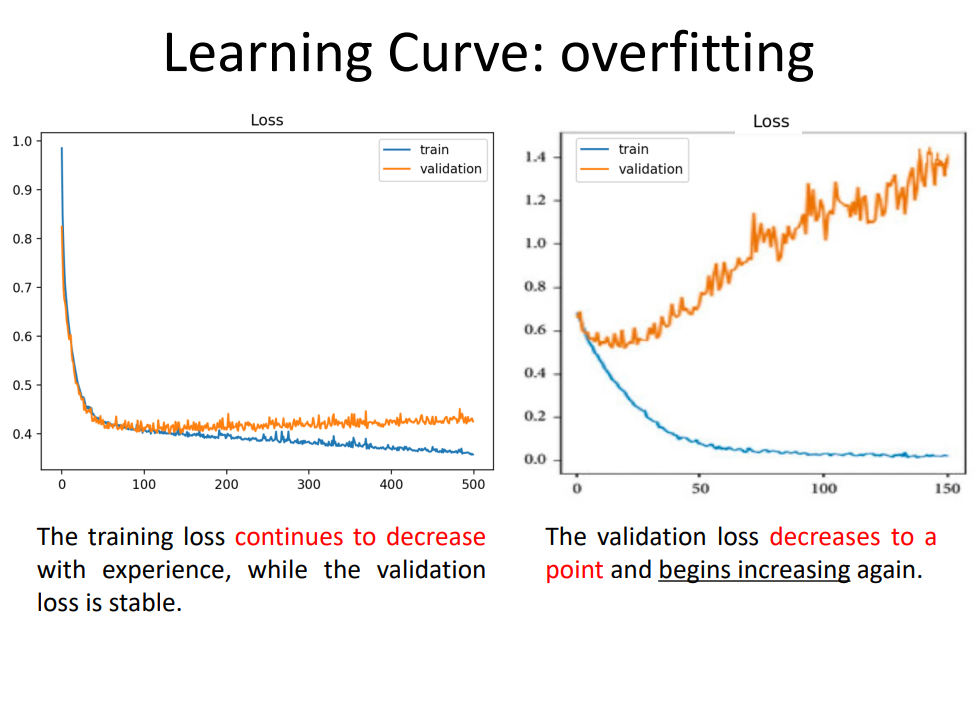
1. **Left Graph:**
   * **Description:** The graph on the left shows a situation where the training loss remains almost constant and very low throughout the training process, as indicated by the flat blue line. The validation loss is similarly flat and is represented by the orange line, which is also constant.
   * **Interpretation:** This pattern is indicative of underfitting, which means the model has not learned the underlying patterns in the training data sufficiently. It is too simplistic and fails to capture the complexity of the data, leading to poor performance on both the training set and the validation set.

Model is too simple

1. **Right Graph:**
   * **Description:** The graph on the right shows a different scenario where the training loss decreases consistently as training progresses, which is depicted by the blue line sloping downwards. The validation loss, shown by the orange line, also decreases but appears to plateau toward the end of the training.
   * **Interpretation:** This is a more typical learning curve, where the model is learning and improving its performance on the training data over time. The training loss is decreasing, which is generally a good sign, but because the validation loss is not improving significantly in the later stages, it may also suggest the onset of overfitting, where the model is beginning to learn the noise in the training data rather than the actual underlying pattern.

The key takeaway from these graphs is that in the context of underfitting:

* The model is too simple and does not capture the complexity of the dataset.
* There is a minimal reduction in loss over epochs, indicating that the model is not improving from additional training.
* The performance on both training and validation sets is poor, suggesting the need for a more complex model or better features to capture the patterns in the data.



1. **Left Graph:**
   * **Description:** The training loss (blue line) decreases significantly at the beginning and then levels off, becoming relatively stable and low. The validation loss (orange line) decreases initially alongside the training loss but then remains stable, not improving much despite further training.
   * **Interpretation:** This graph shows a model that is capable of learning from the training data, as evidenced by the decreasing training loss. The validation loss remains stable after the initial decrease, indicating that the model has reached its capability to generalize to new data. This situation might not necessarily represent overfitting but rather a good fit if the validation loss is acceptably low.
2. **Right Graph:**
   * **Description:** The training loss (blue line) decreases to a very low value and remains low, showing that the model fits the training data very well. However, the validation loss (orange line) decreases to a point but then starts to increase, indicating that the model's performance on the validation set is worsening.
   * **Interpretation:** This is a classic sign of overfitting. The model has learned the training data too well, including noise and outliers, to the extent that its ability to generalize to unseen data is compromised. The increase in validation loss after a certain number of epochs suggests that the model's predictions are becoming less accurate on the validation data, as it is too specialized to the training data.

In the context of overfitting:

* The model performs very well on the training data but poorly on the validation data.
* The increasing validation loss after an initial decrease is a sign that the model is not generalizing well and is capturing noise rather than the underlying data distribution.
* To combat overfitting, one might consider techniques like regularization, getting more training data, reducing the complexity of the model, or using early stopping to halt training before overfitting becomes significant.

Top of Form

Overfitting would be indicated by the validation loss increasing after some point

The first graph in the image you provided depicts a learning curve where the training loss decreases and then stabilizes, while the validation loss remains roughly constant after an initial drop. This pattern suggests that the model is learning from the training data over time and is achieving a stable performance on the validation data.

Now, this pattern might seem like a well-fitting model at first glance, but the context of the image title "Learning Curve: overfitting" and the note about the training loss decreasing continuously while the validation loss is stable can imply two things:

1. **Potential Overfitting (Not Clearly Shown):** If this graph is intended to illustrate overfitting, we would expect to see the validation loss start to increase after a certain point as the model becomes too complex and starts to memorize the training data rather than learning general patterns. However, this increase in validation loss is not depicted in the first graph.
2. **Good Generalization (As Shown):** As the graph is actually shown, it seems to indicate good generalization rather than overfitting because the validation loss is stable and does not increase. The model seems to perform consistently on data it hasn't seen before, which is a desirable outcome in machine learning.

It's important to note that the interpretation of learning curves can be context-dependent, and without more information on the dataset, the model, and other training details, it's challenging to make a definitive assessment. But based solely on the usual interpretation of such graphs, the first graph seems to contradict the overfitting scenario described by the title. If the training loss continues to decrease without an increase in validation loss, it generally suggests good learning progress rather than overfitting.

A graph of loss and loss

Description automatically generated

1. **Left Graph - Unrepresentative Training Set:**
   * **Description:** The training loss (blue line) shows a significant spike at the beginning and then a steady decline, while the validation loss (orange line) decreases less sharply and stabilizes at a higher level than the training loss.
   * **Interpretation:** The annotation suggests that the training set is not representative. This means the data used for training does not adequately capture the characteristics of the broader dataset it's supposed to represent. It might be too small or not diverse enough, leading to a model that doesn't generalize well, as indicated by the higher validation loss.
2. **Right Graph - Unrepresentative Validation Set:**
   * **Description:** The training loss (blue line) drops sharply and then fluctuates at a low level, whereas the validation loss (orange line) is much more volatile and does not show a clear trend of decrease or increase.
   * **Interpretation:** The note indicates that the validation set is not representative. This implies that the data used for validation does not match well with the training data or the real-world data the model will encounter. It could be too small or contain outliers that cause the model's performance to appear more erratic than it actually is, as reflected by the noisy validation loss.

A diagram of a training error

Description automatically generated

1. **Training error increases as more training examples are added**: Ideally, as you add more training data, a well-fitting model should either maintain or decrease its training error, as it learns general patterns. However, if the training error increases high , it suggests that the model is unable to capture the underlying trend even in the data it's being trained on,

indicating that the model is too simple to capture the complexity of the data.

1. **Model is not fitting the data with an appropriate curve**: The model's inability to fit the training data well (evidenced by the increasing training error) suggests that it is underfitting.

This happens when the model does not have enough capacity or complexity to learn the underlying pattern in the data.

1. **Comparable errors in cross-validation and training**: When the training error and the cross-validation (or test) error converge to a value that is above the desired performance (indicated by the horizontal red line), it means that simply adding more data will not likely improve the performance further. Both errors are high and close to each other, which indicates that the model is not just failing to generalize to new data (which would be indicated by high variance), but also failing to learn from the training data itself.
2. **Adding more data does not help**: Generally, adding more data helps to reduce variance, not bias. If adding more data doesn't improve the model's performance, this typically means that the model has a high bias.

To reduce bias, you can try to:

* **Increase the number of features**: This gives the model more information to learn from, which might help it to capture more complexity in the data.
* **Add more complex features**: This could mean adding interaction terms, polynomial features, or leveraging domain knowledge to engineer new features that capture the complexity of the data better.
* **Use a more complex model**: Switching to a more sophisticated algorithm that has a higher capacity to learn more complex patterns might also be necessary.

In summary, this is an indication of high bias because the model is too simple and isn't capable of capturing the underlying trends in the data, even with increasing amounts of training data.

A graph on a white background

Description automatically generated

indication of high variance in a machine learning model. Here's why the slide suggests there is high variance:

1. **Training error grows slowly but remains within the desired range**: This indicates that the model is performing well on the training data and is able to fit that data closely. However, without considering the test error, this is not enough to diagnose high variance.
2. **High variance leads to overfitting and hence high test error**: The test error starts high and remains significantly above the training error as more data is added. This gap suggests that while the model performs well on the training data, it does not generalize well to new, unseen data. This is the hallmark of overfitting, where the model has essentially 'memorized' the training data, including the noise, rather than learning the underlying pattern.
3. **The model is forced to learn more generalized properties**: Ideally, with more data, the model should start learning more generalized patterns that perform well not just on the training data but also on unseen test data.
4. **Adding more examples reduces test error and the gap between the curves closes down**: This suggests that the model is too complex and is capturing noise in the training set. By adding more data, the noise averages out, and the model begins to learn the actual underlying trend, thus improving performance on the test set.

To reduce variance, the slide suggests:

* **Get more data**: More data can help the model to generalize better as it provides more examples to learn from, which can reduce the effect of noise or variance in the training set.
* **Reduce the number of features**: Reducing the complexity of the model by removing some features can prevent the model from fitting to the noise in the training data. This is particularly useful if some features are not relevant to the prediction task and are introducing noise into the model's predictions.

In summary, this scenario indicates high variance because the model is overfitting to the training data, which is evident from the high test error and the large gap between the training and test errors. Reducing variance typically involves simplifying the model or adding more data to improve generalization.

For a **well-fitting model**, as more training data is added, the training error should ideally stay the same or slightly increase. This slight increase can happen because, with more data, there's a higher chance that the model will encounter more variability and potentially more difficult or noisy examples to fit.

Overfitting is typically characterized by a model performing well on the training data but poorly on unseen test data due to the model learning the noise in the training data rather than the underlying pattern. Reducing overfitting usually involves methods to simplify the model or improve its generalization capabilities. Reducing the amount of training data is generally not recommended as a solution to overfitting because:

1. **Less data can exacerbate overfitting**: With less data, the model might have an even easier time memorizing the data rather than learning to generalize, which can actually lead to worse overfitting.
2. **More data helps generalization**: More data usually improves a model's ability to generalize because it provides a more comprehensive representation of the underlying problem, including various edge cases and the overall variability in the input space.

Instead of reducing the data, here are some common strategies to address overfitting:

* **Simplify the model**: Use a less complex model with fewer parameters, or apply techniques like pruning in decision trees.
* **Regularization**: Add a penalty for model complexity to the loss function, such as L1 or L2 regularization.
* **Feature selection**: Reduce the number of input features to eliminate noise or irrelevant data.
* **Cross-validation**: Use cross-validation to ensure the model performs well on multiple subsets of the data.
* **Early stopping**: Stop training before the model becomes too fitted to the training data.
* **Data augmentation**: Increase the size and variety of the training set to improve generalization.
* **Dropout**: For neural networks, dropout layers can help by randomly ignoring a subset of features during training.